AFRL-IF-RS-TR-2007-139
Final Technical Report
May 2007

SOFTWARE WIND TUNNEL (SWiT)

Lockheed Martin Corp

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

NOTICE AND SIGNATURE PAGE

AFRL-IF-RS-TR-2007-139 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/	/s/
WILLIAM E. McKEEVER, JR Work Unit Manager	JAMES A COLLINS, Deputy Chief Advanced Computing Division Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
MAY 2007	Final	Aug 06 – Jan 07

4. TITLE AND SUBTITLE

SOFTWARE WIND TUNNEL (SWiT)

5a. CONTRACT NUMBER
FA8750-06-C-0205

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
63781D

6. AUTHOR(S)

Richard W. Buskens, Patrick J. Lardieri, Bennett C. Watson, Jennifer Lautenschlager and Douglas C. Schmidt

5d. PROJECT NUMBER
SSTT

5e. TASK NUMBER
LM

5f. WORK UNIT NUMBER
06

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Lockheed Martin Corp
Advanced Technology Laboratories
3 Executive Campus-6th Floor
Cherry Hill NJ 08002

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/IFTC
525 Brooks Rd
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2007-139

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA# 07- **240**

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Technology transition is a serious problem plaguing the DoD today. This report summarizes the activities undergone and results developed by the Lockheed Martin Software Wind Tunnel (SWiT) team to develop a concept of operations (CONOPS) and system architecture to address AFRL's Systems and Software Test Track objectives of providing an open collaborative research and development environment to demonstrate, evaluate and document the ability of novel tools, methods, techniques and run-time technologies to yield affordable and more predictable production of software intensive systems.

15. SUBJECT TERMS
Collaboration portal, experimentation testbed, concept of operations, system architecture

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			William McKeever
U	U	U	UL	38	19b. TELEPHONE NUMBER (*Include area code*)

Preface

The Lockheed Martin Advanced Technology Laboratories' Team developed this document for Air Force Research Laboratories (AFRL) under contract FA8750-06-C-0205 in order to fulfill the objective of providing a final technical report for the "Software Wind Tunnel" (SWiT) implementation of the Systems and Software Test Track (SSTT).

Table of Contents

List of Figures

List of Tables

1 Summary

Software is the weak link in many DoD systems. These software-*intensive systems (SIS)* demonstrate the limits of our current technology for software specification, development and testing. As a result, the greatest risks to meeting the operational capabilities, schedule and cost in most large-scale DoD systems can be traced to software producibility issues. New technologies must be developed or we will find our ability to create new DoD systems fundamentally limited. A critical aspect orthogonal to the actual software producibility technology being produced or provided is the difficulty of introducing advanced development tools and technologies into DoD acquisition programs.

To help resolve the specific challenge of technology transitioning, the Lockheed Martin team has developed a concept of operations (CONOPS) and a system architecture for a Software Wind Tunnel (SWiT), a collaboration environment and shared experimental testbed aimed at bringing together researchers, developers, and domain experts from different communities to de-fragment the knowledge necessary to achieve Software-Intensive Systems Producibility Initiatives (SISPI) technology transition. A formal two-day workshop and informal but regular engagements among various Lockheed Martin SWiT team members and its consultants helped guide the development and evolution of the SWiT CONOPS and architecture, leveraging Double Helix methodologies (used by DARPA's Command Post of the Future) to co-evolve SWiT technology and its application to DoD software producibility issues. In addition, members of the SWiT team had highly valuable bi-weekly interactions with the AFRL team, and frequent team meetings.

Our team consisted of researchers from Lockheed Martin Advanced Technology Laboratories, Lockheed Martin Aeronautics Company and Vanderbilt University, with additional consultation provided by the Office of the Secretary of Defense (OSD), AFRL, the Software Engineering Institute (SEI), University of Maryland, University of Utah and program engineers from other Lockheed Martin business organizations.

2 Introduction

2.1 Problem Summary

Technology transition is a serious problem plaguing the DoD today. Figure 1 depicts the DoD's Technology Readiness Levels (TRL) in relation to the types of funding (numbered boxes) targeted at particular levels [1]. The DoD general strategy for technology transition relies on different organizations, e.g., DARPA, Armed Services Laboratories, and the Services themselves, to fund technology efforts in the "Science & Technology", "Research & Engineering", Procurement, and O&M areas.

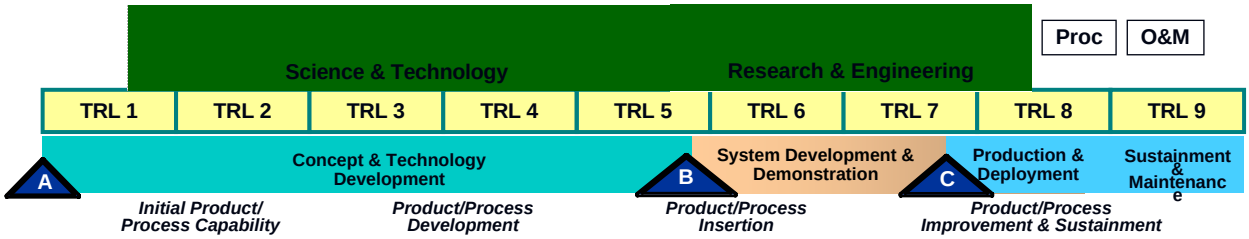


Figure 1. DoD Technology Maturity Lifecycle

The expectation is that promising work funded by one organization early in the lifecycle, e.g., a DARPA program focusing on Science and Technology, will be perpetuated by another organization in the following phase, e.g., an AFRL program on Research & Engineering. The DoD has over 100 separate organizations [2] engaged in technology transfer. However, technology transition today is a largely *ad hoc* process, as it relies on government sponsors, industry engineers, and university researchers to be aware of joint interests and self-organize into collaborative teams. The current strategy and number of organizations involved make it hard for knowledgeable people to collaborate and plan for transition.

This poor collaboration among people working across the technology maturity lifecycle has created a "valley of disappointment" where DoD programs fail to adopt advanced technologies, regardless of their inherent promise, and software producibility problems are encountered repeatedly across programs. To exacerbate the problem further, the "landing path" is typically not DoD program engineers but an organization responsible for sustaining the technology (e.g., a commercial vendor).

Figure 2 shows SISPI transition processes conducted today. In today's processes, *government* personnel working DoD acquisition programs and *government* personnel working research programs collaborate to define a SISPI problem and research agenda. The challenge problem is written into the research program's BAA and performers then bid specific transition plans.

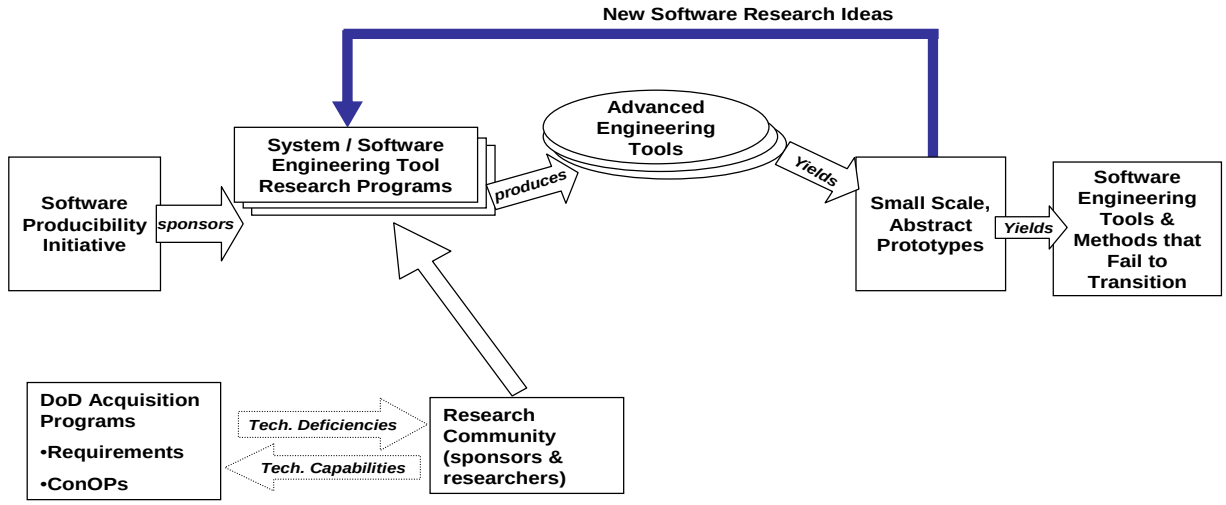


Figure 2. SISPI Technology Transition Using Current Methods

SISPI Researchers awarded contracts typically have no relationship with Program Engineers in the program or domain from which the challenge problem is drawn. The SISPI researchers do their best to understand and incorporate specific knowledge about the Program Engineer's problem domain but getting detailed information is in general difficult, and even more difficult when classification and ITAR issues are involved. SISPI Researchers thus have little choice but to design and conduct experiments that are abstract and typically small-scale representations of the real challenge problem. These results may show the promise of the new technology but leave a large "credibility gap" in the minds of Program Engineers about how the results will transition into the real problem domain. The work to overcome this "credibility gap" is typically left to other programs that are funded by other DoD agencies or the acquisition program. The ultimate success or failure of technology transition thus depends on the *ad hoc*, opportunistic transition process describe above, where serendipity of the right people being in the right positions is the primary enabler for success.

The degree of Program Engineer engagement in research programs varies with individual SISPI Program Managers. In an effort to increase the likelihood of transition some SISPI Program Managers fund Open Experimental Platforms (OEP) to structure engagements between SISPI Researchers and Program Engineers early and throughout the program. This practice has resulted in research that is better targeted at real program challenges. To overcome the credibility gap, however, it is still left to other programs/people to address the problem of demonstrating the technology in the actual environment.

2.2 Systems and Software Test Track Requirements

SWiT aims to address the above-described technology transition problem. AFRL's Systems and Software Test Track (SSTT) program explicitly solicited technology solutions targeted to significantly improve technology transition within the DoD. In their words:

> "…the objective of the Systems and Software Test Track is to provide an open collaborative research and development environment to demonstrate, evaluate, and document the ability of novel tools, methods, techniques, and run-time technologies to yield affordable and more predictable production of software intensive systems."

To achieve the above objective requires that the following four key requirements be satisfied:

R1. There must be an ability to define and collaborate around challenge problems.

R2. There must be an ability to define and collaborate around candidate solutions to challenge problems.

R3. There must be an ability to define, conduct and collaborate around experiments related to challenge problems and candidate solutions.

R4. There must be an ability to transition technologies across the software producibility spectrum and across operational domains.

2.3 Report Outline

The remainder of this report is structured as follows. Section 3 presents assumptions, the methods and procedures used in defining SWiT. Section 4 discusses our original and revised SWiT visions, presents an overview of SWiT, summarizes the SWiT CONOPS and architecture, discusses SWiT operational policies and constraints, presents SWiT usage scenarios and briefly discusses prototyping and validation activities performed. Section 5 provides concluding remarks, and Section 6 presents our recommendations on future SSTT activities.

3 Assumptions, Methods and Procedures

3.1 Assumptions

Table 1 summarizes the areas of concern that the DoD's Software Intensive Systems Producibility Initiative (SISPI) will address: Technology Transition, System Design & Development, and System Test & Certification.

	Technology Transition	System Design & Development	System Test & Certification
Key Challenge	Broker experimental collaborations between emerging advanced technology solutions and open development program challenges to accelerate technology transition.	Provide system design and software development tools, methods, and services for producing large-scale DoD software intensive systems reliably and cost effectively.	Provide integration, test, and certification tools, methods, and services for cost effectively assuring functional and quality of service properties of large-scale DoD software intensive systems.
Driving Actors	Technology Researchers Development Program Software Architects and System Engineers	Development Program System Engineers Development Program Software Architects and Engineers	Development Program System Engineers Development Program Integration & Test Engineers

Table 1. Summary of Challenges in the DoD's Software Intensive Systems Producibility Initiative (SISPI)

AFRL's SSTT program addresses the Technology Transition problem area with a focus on enabling the transition of novel system design, development, test and certification technologies developed under other SISPI programs.

3.2 Methods and Procedures

We developed our SWiT Concept of Operations (CONOPS) and SWiT System Architecture through a single workshop as well as regular engagements with a multi-disciplinary group of representative stakeholders, including university researchers, DoD acquisition program engineers, experts from academia and government agencies, DoD personnel from the acquisition and Special Programs Office communities, and DoD problem domain experts from Lockheed Martin. Internally, we drew from ongoing programs including the Software Technology Initiative (STI), Horizontal Integration, JSF, F-22, SIBRS High, DD(X) Land Attack Destroyer program, Next Generation Air Operations Center, Aegis Open Architecture, Navy Open Architecture, Future Combat System (FCS), Global Information Grid (GIG) and others. Activities to develop/evolve the SWiT CONOPS and architecture were guided by the Double Helix methodology developed by ISX corporation (now a component of Lockheed Martin Advanced Technology Laboratories) and used by DARPA's Command Post of the Future (CPOF), which has recently transitioned to Iraq. The Double Helix methodology involves the co-evolution of technology and its application to DoD systems, thus ideally suited to the needs of SWiT to identify the driving problems impeding technology transition into DoD software-intensive systems. Frequent interactions with AFRL, especially in the latter half of the program, proved highly valuable in refining the SWiT CONOPS and architecture to help ensure that AFRL's objectives were being met.

To aid in maintaining close participation among the SWiT team members, and to invite feedback from participating external experts, SWiT discussions and documentation were posted on a wiki site at Vanderbilt University.

4 Results and Discussion

4.1 Original SWiT Vision

Our original SWiT vision focused on a specific class of software producibility problems, namely those associated with predicting the performance of complex distributed systems architectures with multiple competing performance requirements and multiple platform technology infrastructures. The vision was driven by the complex nature of the problem, observed impacts across multiple DoD programs (e.g., F-22 and DDG-1000), limited transition of existing research solutions due to the difficulty of demonstrating applicability to real systems (e.g., end-to-end schedulability analysis), open research challenges in forming a general system execution modeling and performance prediction theory, plus the team's innovative technologies and research (e.g., Emulab, CUTS, Thimble).

However, this original vision did not satisfy requirement R4: the ability to transition technologies across the software producibility spectrum and across operational domains (refer to Section 2.2). Specifically, in addition to tools and technologies for system execution modeling, static and dynamic analysis methods, program transformation techniques, software design tools, programming languages, collaboration tools and other software producibility research areas must equally be supported by SWiT. This key insight was gained during our workshop, held in early October 2006, The workshop participants recognized that initial prototypes of experimental testbeds to showcase and validate candidate solutions to challenge problems may need to focus on specific domains (e.g., avionics).

4.2 Revised SWiT Vision

We thus revised our vision of SWiT, broadening it to more prominently feature a collaboration environment for bringing program engineers and SISPI researchers together. Figure 3 shows our new vision for SISPI technology transition using SWiT.

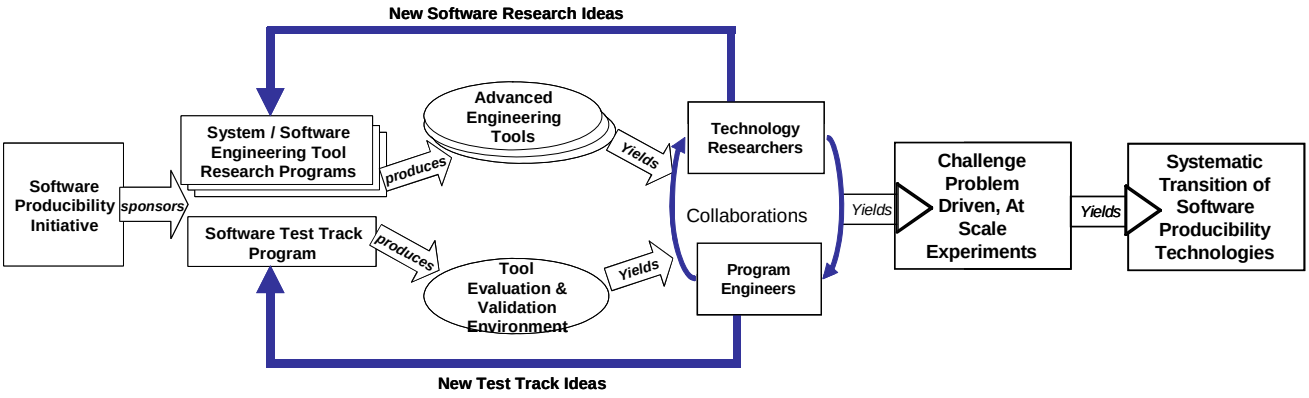


Figure 3. SISPI Technology Transition Using SWiT

That is, our revised SWiT vision provides methods and technologies that create a common meeting house for program engineers and technology researchers to discover joint interests and form collaborations. This collaboration spans all software producibility research areas. SWiT itself will contain testbed resources and tools specialized to particular operational domains (e.g., avionics). The anticipated result is that challenge problem-driven at-scale experiments can be defined and conducted to prove/disprove suitability of particular software producibility technolo-

gies, which leads to a more *systematic process for transitioning software producibility technologies*.

The next section presents a high-level overview of SWiT.

4.3 Overview

SWiT is a collaboration environment and shared experimental testbed aimed at bringing together researchers, developers, and domain experts from different communities to de-fragment the knowledge necessary to achieve SISPI technology transition. Figure 4 depicts a context diagram to illustrate this, and highlights several logical roles of collaborating participants: Challenge Problem Providers (for providing challenge problems), Candidate Solution Providers (for providing candidate solutions to challenge problems), Experimenters (for running experiments and collecting experimental results), Collaborators (involved in collaborating on challenge problems, candidate solutions, and/or experiments), and SWiT Administrators (responsible for administering the SWiT infrastructure). Various community participants may play one or more of these logical roles. Overall, SWiT will enable program engineers from DoD acquisition programs to interact with SISPI researchers from industry labs or universities to define, discover and evaluate technology transition opportunities. SWiT will also enable SISPI program managers to discover new SISPI research challenges and to measure transition success or failure of their program research.

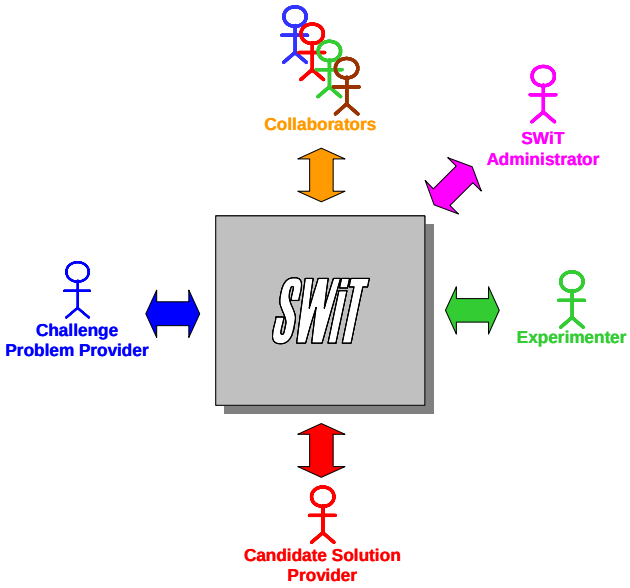


Figure 4. SWiT Context Diagram

Conceptually, SWiT provides hardware, software and process infrastructure to enable the SSTT objective; i.e., to provide an open collaborative research and development environment to demonstrate, evaluate, and document the ability of novel tools, methods, techniques, and run-time technologies to yield affordable and more predictable production of software intensive systems. The hardware may be an aggregation of centralized resources, distributed resources connected via internet and local (stand-alone) resources. The unifying element for SWiT will be the software suite. The software will enable distributed, interactive collaboration among the users, tool integration, performance measurement and data collection, and a repository for SWiT artifacts.

It is envisioned that the software resides at a centralized facility and be available for download into local, standalone environments.

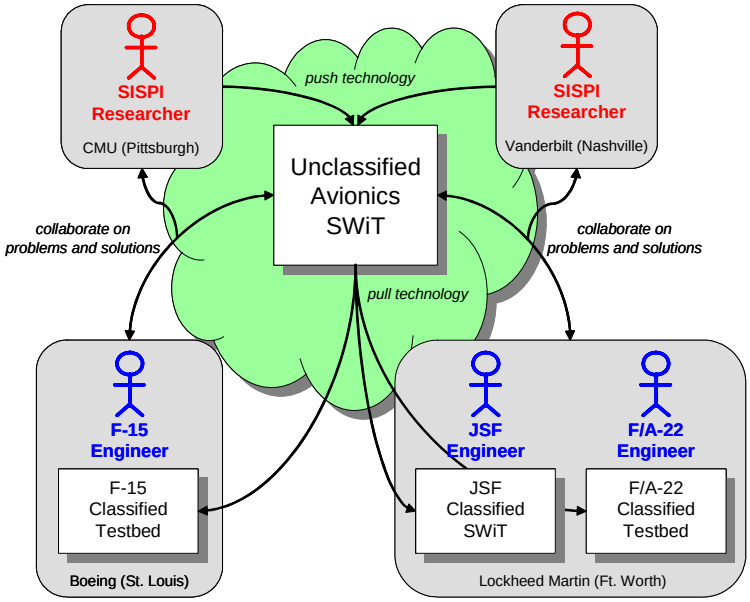


Figure 5. Notional SWiT System View

We envision that there will be multiple SWiT instances, each dedicated to enabling collaboration among a specific community, e.g., distributed real-time embedded weapons systems or enterprise C2/C4ISR information systems. Given the distributed nature of users in these communities, SWiT will be a remotely accessible system with visual and text interfaces. It will be hosted and operated by the DoD or a designated contractor, but will be accessible from the public Internet and will only host unclassified data. Since classified evaluations will be part of the technology transition process, users of SWiT may download its tools to conduct these experiments within their classified program environments. Likewise, users can stand up a classified SWiT instance within their classified program environment. Figure 5 summarizes this system view.

We now discuss the SWiT concept of operations (CONOPS).

4.4 SWiT CONOPS

4.4.1 Functional Architecture Overview

Figure 6 depicts the SWiT functional architecture.

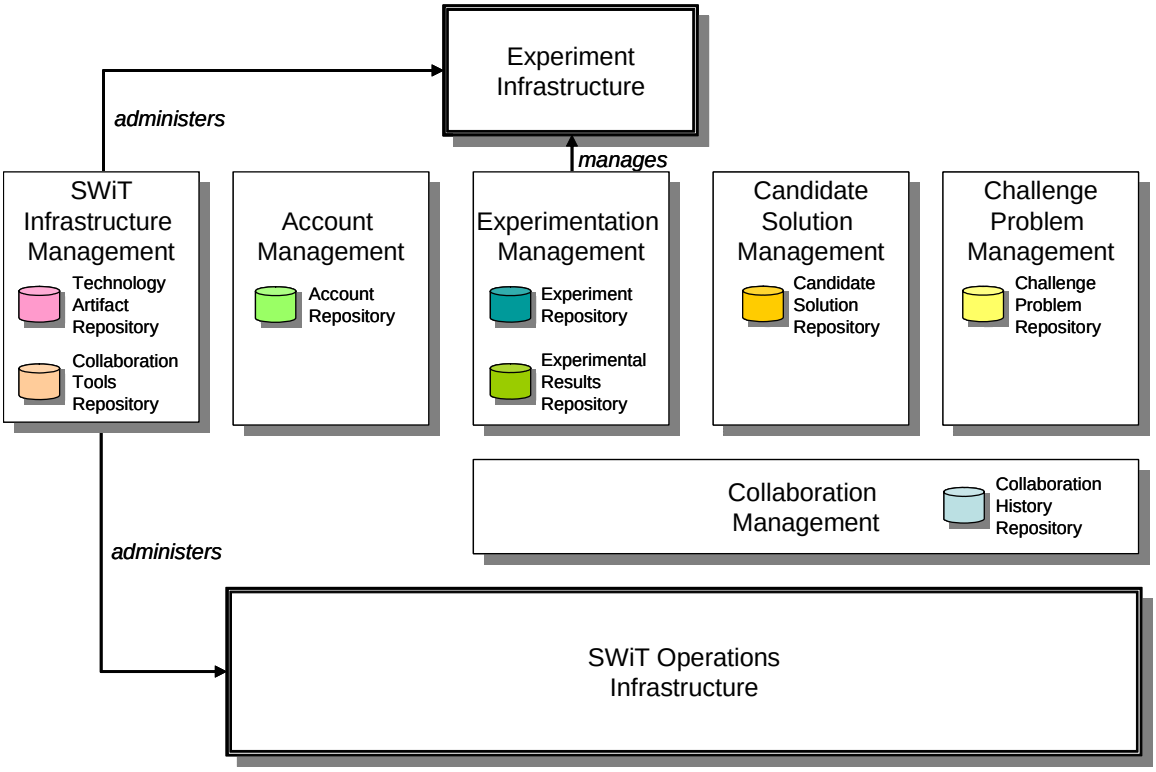


Figure 6. SWiT Functional Architecture

SWiT comprises two functional infrastructure components. These infrastructure components and their purpose in SWiT are:

- **SWiT Operations Infrastructure**: hardware and software that enables users to effectively utilize SWiT. The hardware would likely be general purpose server machines that may host, for example, web server software. Other software might include value-add tools that simplify the user's interaction with SWiT. An example of such tools might be an enhanced version of Skype software whereby phone calls placed using this software via SWiT will automatically log the call as part of the collaboration history associated with a challenge problem.

- **Experiment Infrastructure**: hardware and software for experimentation to test out hypotheses, reproduce challenge problems, and test and evaluate candidate solutions to challenge problems.

More importantly, SWiT consists of a set of logical functional components. These components and their purpose in SWiT are:

- **Challenge Problem Management**: manages information about challenge problems. Allows challenge problems to be created and edited, searched, archived and endorsed. Data related to these challenge problems is stored in the Challenge Problem Repository.

- **Candidate Solution Management**: manages information about candidate solutions to challenge problems. Allows candidate solutions to be created, edited and searched. Ensures that candidate solutions are associated with challenge problems stored in SWiT. Data related to these candidate solutions, excluding the technology itself, is stored in the Candidate Solution Repository.

- **Experimentation Management**: manages information about and execution of experiments. Allows experiments to be created, edited, searched, and executed. Allows experimental results to be published and downloaded. Supports uploading and downloading of technology and experimental results into and out of SWiT. Data related to the experiments is stored in the Experiment Repository. Experimental results and associated relevant information is stored in the Experimental Results Repository.

- **Collaboration Management**: manages information related to collaborations among SWiT users. Provides capabilities to establish new collaborations and to create, edit, search and review interactions among collaborators (e.g., captures an e-mail or voice exchange among a pair of collaborators) and maintains a collaboration history. These interactions are captured in the Collaboration History Repository and may be associated with challenge problems, candidate solutions, experiments and experimental results.

- **Account Management**: manages SWiT user accounts and account related issues. User account and related information is stored in the Account Repository.

- **SWiT Infrastructure Management**: administers both the SWiT Operations Infrastructure and the SWiT Experiment Infrastructure. Tools provided as part of the SWiT infrastructure to facilitate collaboration are stored in the Collaboration Tools Repository. Candidate solution technology made available for general use is stored in the Technology Artifact Repository.

Each logical component maintains one or more repositories. The repositories and their logical contents are:

- **Challenge Problem Repository**: maintained by the Challenge Problem Management functional component, this repository contains data for each challenge problem, including:

    - The challenge problem domain. Challenge problem domains, in this context, are limited to software producibility domains. Candidate challenge problem domains are: requirements engineering, architecture, design, code, test and deployment. Note that a particular challenge problem may span multiple challenge problem domains.

    - A set of relevant system characteristics – e.g., real-time system, embedded system, etc.

    - Challenge problem description. Free text input is anticipated to be the most common means of entering the challenge problem description. However, alternative input forms, e.g., restrictive grammars implemented through a collection of radio buttons on a web page, should also be supported. The challenge problem description may be high-level or low-level.

    - Optionally, linkages between this challenge problem and others already in the SWiT challenge problem database.

    - Optionally, development and execution platform characteristics – e.g., Windows XP development platform, QNX 6.1 execution platform.

    - Optionally, any development tools used.

    - Optionally, any source code or binary code examples (can be used both to illustrate the problem and evaluate candidate solutions).

- **Candidate Solution Repository**: maintained by the Candidate Solution Management functional component, this repository contains a description of the proposed solution, plus other useful attributes – e.g., type of solution (e.g., process, tool, technology), domain of applicability, development platform, run-time platform, linkage to related solutions, linkage to challenge problems, linkage to assets in the Technology Asset Repository (where actual technology/tools may be stored), etc.

- **Experiment Repository**: maintained by the Experiment Management functional component, this repository contains information about the following:

    - Links to a specific challenge problem or set of challenge problems. That is, which problem(s) the experiment intends to target.

    - Links to a candidate solution or set of candidate solutions. Some experiments will need to be tailored to specific candidate solutions, while other experiments may not be dependent upon any specific candidate solution.

    - Details about the desired experimental configuration, e.g., the number and types of hardware nodes, and platform characteristics (e.g., operating system, device drivers, communications protocol stacks, compilers and linkers, software libraries, etc.).

    - Details about software tools and technologies either necessary for or to be used in the experiment (e.g., Eclipse IDE with specific plug-ins). These tools and technologies would not be considered a part of the "standard" experiment infrastructure, but would be brought into the experiment infrastructure specifically for the experiment (e.g., a specially instrumented version of an operating system).

    - Any input data needed for the experiment (e.g., source code, configuration data, etc.).

- **Experimental Results Repository**: maintained by the Experiment Management functional component, this repository contains results of executed experiments, including details on how to analyze/interpret these results. The results and their interpretation are expected to be experiment-specific.

- **Collaboration History Repository**: maintained by the Collaboration Management functional component, this repository contains records of all interactions around challenge problems, candidate solutions, and experiments. For a specific challenge problem, candidate solution, or experiment each record contains a date/timestamp, a brief (short phrase) summary description of the interaction, a list of collaborators involved in the interaction, plus the interaction itself (an e-mail message, a voice call, etc.).

- **Technology Artifact Repository**: maintained by the SWiT Infrastructure Management functional component, this repository contains technology artifacts (e.g., software producibility tools) that are made available to all SWiT users who configure and run experiments. A brief description of the technology, platforms on which the technology may be applied, documentation to help SWiT users learn and use the technology, as well as the technology itself are stored in this repository.

- **Collaboration Tools Repository**: maintained by the SWiT Infrastructure Management functional component, this repository contains tools that aid in the collaboration process itself. Specific examples of such tools include: NetMeeting, Skype, etc., along with any extensions to these tools that simplify incorporating collaborative interactions into SWiT (e.g., by automatically storing a voice call associated with a particular challenge problem in the collaboration history for the challenge problem). A brief description of the tool, platforms on which the tool may be used, documentation to help SWiT users learn and use the tool, as well as the tool itself are stored in this repository

- **Account Repository**: maintained by the Account Management functional component, this repository contains account information for all SWiT users. The account information includes:

    - Organization name, name and contact information (e.g., e-mail address, postal address, phone number, fax number) for a primary point of contact, a list of contributing members of the organization, etc.

    - A username, unique for each SWiT user.

    - A password.

    - A series of challenge questions and their responses (so that passwords can be reset if the user forgets his/her password).

4.4.2 Physical Personnel and Logical Actors

The personnel that we envision will be involved in SWiT fall into the following five categories:

- SISPI Researcher: performs research and, through this research, provides technology that aims to address challenge problems.

- Program Engineer: executes on DoD programs.

- DoD Program Manager: oversees DoD programs.

- Industry Partner: matures technology provided by SISPI Researchers to harden it for use by Program Engineers.

- Administrator: administers the SWiT operations and experiment infrastructure. Not specifically associated with traditional/current technology transition activities.

In the context of SWiT, each of the above classes of “physical” personnel may take on various logical roles. For convenience, we distinguish the various roles played by members of the above physical user classes by introducing a set of logical actors. Figure 7 identifies these logical actors and their associated roles.

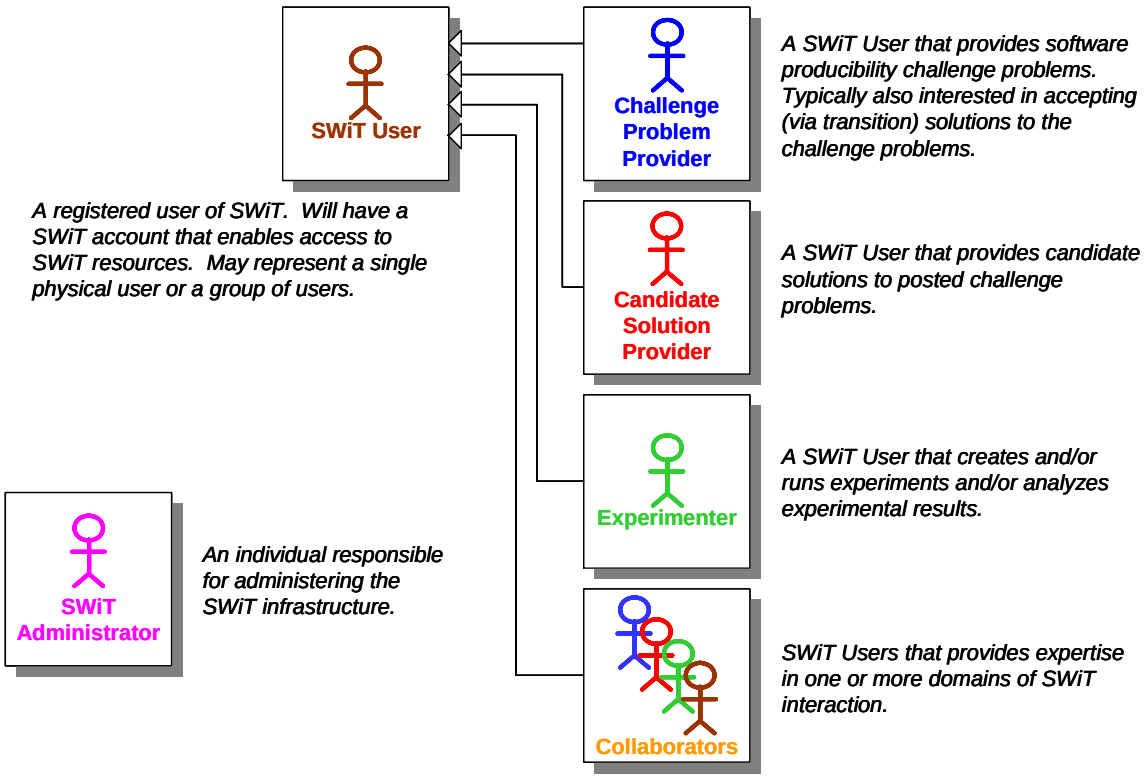


Figure 7. SWiT Logical Actors and Their Roles

In practice, the physical personnel will map to one or more of the above logical actors, and all will take on the logical role of SWiT User. For example, a Program Engineer will likely be a Problem Provider and a Collaborator. SISPI Researchers will often take on the roles of Candidate Solution Provider, Experimenter and Collaborator. This does not prevent physical personnel for taking on a subset of these roles – e.g., a SISPI Researcher might only participate as a Collaborator in some instances.

Figure 8 shows the relationship of the SWiT logical actors to the SWiT functional architecture components.

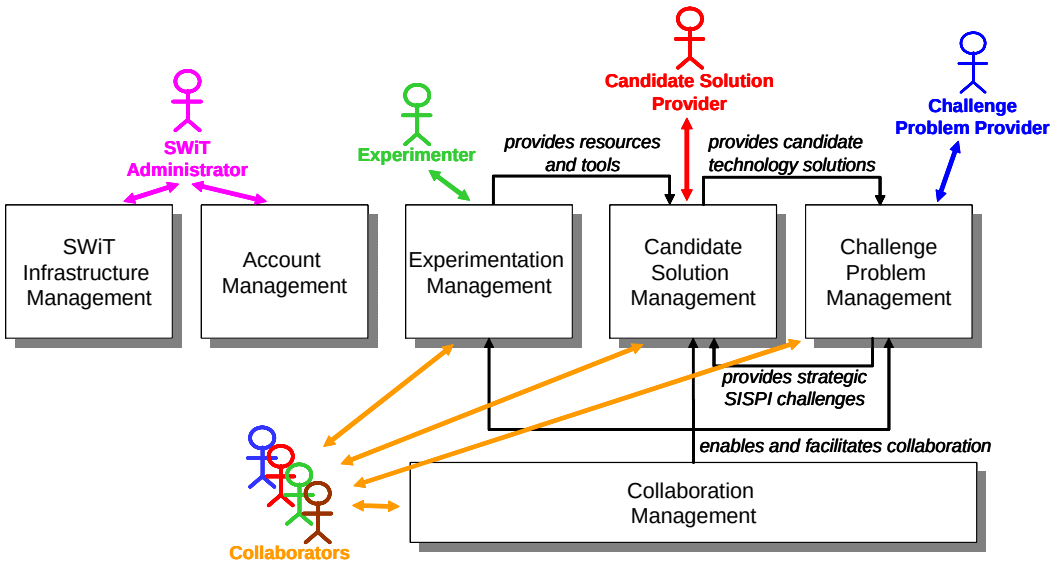


Figure 8. Logical Actor Relationship to SWiT Functional Architecture Components

4.4.3 Use Case Overview

The SWiT CONOPS is broken down into six categories of use cases, one for each functional component. In the subsections that follow, we briefly summarize the use cases associated with each functional component.

4.4.3.1 Account Management Use Cases

Table 2 summarizes the account management capabilities of SWiT, which allow SWiT Users and SWiT Administrators to manage account creation and termination, edit account properties (e.g., update contact information, change password), and log into and out of SWiT. These capabilities are intended to be similar to those provided by other collaboration portals (e.g., wiki, BCSW, etc.)

	Use Case Name	Description
1	Create SWiT Account	Provide a mechanism by which interested parties can register with SWiT. Once registered, interested parties can access the SWiT operations and experiment infrastructure, interact with other SWiT members, etc.
2	Delete SWiT Account	This operation allows SWiT Users to be removed from the SWiT system.
3	Edit SWiT Account Information	Allows a registered and logged-in SWiT User to edit account information (e.g., update point of contact information, modify list of contributing members, change password, etc.).
4	Login to SWiT	Provide a registered SWiT User the ability to login to his/her account, giving them access to SWiT resources.
5	Reset Password	Allows a SWiT User to reset their password (in case they forget their password).
6	Logout	Logs a SWiT User out of the SWiT system.

Table 2. Account Management Use Case Summary

4.4.3.2 Collaboration Management Use Cases

Table 3 summarizes SWiT use cases related to managing collaborator participation among SWiT Users in their various roles. We intend that while collaboration is to be promoted and supported wherever possible, a mechanism to reject undesired collaboration should be provided (e.g., to minimize the impact of security attacks). Hence, to engage a new collaborator, the collaborator must explicitly be invited to participate or must explicitly request an invitation to participate.

Specific collaboration activities associated with challenge problems, candidate solutions to challenge problems, and experimentation intended to prove in candidate solutions to challenge problems are captured in their respective sections below.

	Use Case Name	Description
1	Invite Collaborator	A Collaborator (i.e., Problem Provider, Candidate Solution Provider, Experimenter, a SWiT User) invokes this use case to invite another SWiT User to participate in an ongoing collaboration.
2	Accept Collaboration Invitation	Invoked by a SWiT User who receives a collaboration invitation and wishes to participate in the collaboration.
3	Reject Collaboration Invitation	Invoked by a SWiT User who receives a collaboration invitation and does not wish to participate in the collaboration.
4	Request Invitation to Collaborate	Invoked by a SWiT User who is interested in collaborating with other SWiT Users already collaborating to address a particular challenge problem.
5	Reject Collaborator	Complement of Invite Collaborator. Issued by one of the Collaborators associated with a specific challenge problem, its candidate solutions or its experiments to respond negatively to an unsolicited request by a SWiT User to participate in the collaboration.

Table 3. Collaboration Management Use Case Summary

4.4.3.3 Challenge Problem Management Use Cases

Table 4 summarizes SWiT use cases associated with managing challenge problems. These use cases allow challenge problems to be created and edited, searched, archived and endorsed. They also allow collaborations associated with challenge problems to be created, deleted, edited and viewed. Additionally, SWiT Users can register to be asynchronously notified of changes to challenge problems and their collaboration histories.

	Use Case Name	Description
1	Create Challenge Problem	Create a new challenge problem.
2	Delete Challenge Problem	Remove a challenge problem from the SWiT challenge problem repository.
3	Edit Challenge Problem	Edit or refine an existing challenge problem.
4	Endorse Challenge Problem	Express interest in and/or support of a challenge problem that was entered into the SWiT challenge problem repository by a different SWiT User.
5	Reject Challenge Problem	Reject a proposed challenge problem.
6	Close Challenge Problem	Close an existing challenge problem. No further collaboration around the challenge problem is permitted.
7	Reopen Challenge Problem	Re-open a challenge problem that has previously been closed. Collaborations related to the challenge problem are again permitted.
8	Archive Challenge Problem	Archives closed challenge problem .
9	Browse Existing Challenge Problems	Browse the set of existing challenge problems.
10	Search for Challenge Problem	Report challenge problems that match specific search criteria.
11	Create Challenge Problem Collaboration Item	Create a new collaboration item associated with a specific challenge problem. Viewable by all parties interested in collaborating on or observing activities related to the specific challenge problem.
12	Delete Challenge Problem Collaboration Item	Deletes a collaboration item from the SWiT challenge problem collaboration history repository.
13	Edit Challenge Problem Collaboration Item	Modify/update an existing collaboration item.
14	View Challenge Problem Collaboration History	View the collaboration history of a specific challenge problem.
15	Register to be Notified of New Challenge Problem	Register to receive a notification when new challenge problems are created.
16	Cancel Registration to be Interested of New Challenge Problem	Invoked to stop receiving notifications when new challenge problems are created.
17	Register Interest in a Specific Challenge Problem	Register to receive a notification when updates are made to a challenge problem, its attributes, collaboration history, list of candidate solutions, etc.
18	Cancel Registration of Interest in a Specific Challenge Problem	Invoked to stop receiving notifications of changes to challenge problem related information.

Table 4. Challenge Problem Management Use Case Summary

4.4.3.4 Candidate Solution Management Use Cases

Table 5 summarizes SWiT use cases associated with managing candidate solutions to challenge problems. These use cases allow candidate solutions to challenge problems to be created, edited and searched. They also allow collaborations associated with candidate solutions to challenge problems to be created, deleted, edited and viewed. Additionally, SWiT Users can register to be asynchronously notified of changes to candidate solutions to challenge problems and their collaboration histories.

	Use Case Name	Description
1	Propose Candidate Solution to Challenge Problem	Propose a solution that may be useful in solving specific challenge problems.
2	Delete Candidate Solution to Challenge Problem	Remove a candidate solution from the SWiT candidate solution repository.
3	Edit Candidate Solution to Challenge Problem	Edit or refine an existing candidate solution to a challenge problem.
4	Notify Intent to Adopt Solution	Indicate intent to adopt the candidate solution.
5	Close Candidate Solution to Challenge Problem	Close an existing candidate solution to a challenge problem. No further collaboration around the candidate solution is permitted.
6	Reopen Candidate Solution to Challenge Problem	Re-opens a challenge problem that has previously been closed.
7	Browse Existing Candidate Solutions to Challenge Problem	Browse the set of existing candidate solutions.
8	Search for Candidate Solution to Challenge Problem	Report candidate solutions that match specific search criteria.
9	Create Candidate Solution Collaboration Item	Create a new collaboration item associated with a specific candidate solution. Viewable by all parties interested in interacting around the specific candidate solution.
10	Delete Candidate Solution Collaboration Item	Delete the item from the SWiT candidate solution collaboration history repository.
11	Edit Candidate Solution Collaboration Item	Modify/update an existing collaboration item.
12	View Candidate Solution Collaboration History	View the collaboration history of a specific candidate solution.
13	Register to be Notified of New Candidate Solution	Register to receive a notification when new candidate solutions are created.
14	Cancel Registration to be Notified of New Candidate Solution	Invoked to stop receiving notifications when new candidate solutions are created.
15	Register Interest in Candidate Solution to Challenge Problem	Register to receive a notification when updates are made to a candidate solution, its attributes, collaboration history, etc.
16	Cancel Registration of Interest in Candidate Solution to Challenge Problem	Invoked to stop receiving notifications of changes to candidate solution related information.

Table 5. Candidate Solution Management Use Case Summary

4.4.3.5 Experimentation Management Use Cases

Table 6 summarizes SWiT use cases associated with managing experimentation related activities in SWiT. These use cases allow experiments to be created, edited, searched, started and stopped, and experimental results to be posted, analyzed and downloaded. They also allow collaborations associated with experiments to be created, deleted, edited and viewed. Additionally, SWiT Users can register to be asynchronously notified of changes to experiments, experimental results and their collaboration histories.

	Use Case Name	Description
1	Create Experiment	Create a new experiment that is intended to be executed on the SWiT experiment infrastructure.
2	Delete Experiment	Remove an experiment from the SWiT system.
3	Update Experiment	Edit or refine an existing experiment.
4	Browse Experiments	Browse the set of existing experiments.
5	Search Experiments	Report experiments that match specific search criteria.
6	Create Experiment Collaboration Item	Create a new collaboration item associated with a specific experiment. Viewable by all parties interested in interacting around the specific experiment.
7	Delete Experiment Collaboration Item	Delete the item from the SWiT experiment collaboration history repository.
8	Edit Experiment Collaboration Item	Modify/update an existing experiment collaboration item.
9	View Experiment Collaboration History	View the collaboration history of a specific experiment.
10	Begin Experiment	Start an experiment. Experimental resources are allocated and configured.
11	Post Experimental Results	Post the results of a completed experiment.
12	Analyze Experimental Results	Examine the results of an experiment. All relevant information will previously have been posted.
13	Download Experimental Results	Download the results of an experiment, for possible offline analysis.
14	End Experiment	Indicate that an experiment has ended. Resources allocated and configured for the experiment are unallocated by SWiT.
15	Register Interest in New Experiments	Register to receive a notification when new experiments are created.
16	Cancel Registration of Interest in New Experiments	Invoked to stop receiving notifications when new experiments are created.
17	Register Interest in Specific Experiment	Register to receive a notification when updates are made to an experiment, its attributes, collaboration history, etc.
18	Cancel Registration of Interest in Specific Experiment	Invoked to stop receiving notifications of changes to experiment related information.

Table 6. Experimentation Management Use Case Summary

4.4.3.6 SWiT Infrastructure Management Use Cases

Table 7 enumerates the use cases associated with managing the SWiT operations and experiment infrastructures for the purpose of technology transition. These use cases enable new tools and technology to uploaded, downloaded, posted and removed. They also permit new SWiT infrastructure instances to be registered and unregistered.

	Use Case Name	Description
1	Upload Tool/Technology	Upload new collaboration tools or new software producibility tools and technology to be uploaded to SWiT.
2	Download Tool/Technology	Download collaboration tools or software producibility tools and technology from SWiT for private use outside of SWiT.
3	Post Tool/Technology	Make new tools or technology available for the broader SWiT community to use. The tools or technology must have previously been uploaded via use case Upload Tool/Technology.
4	Remove Tool/Technology	Decommission or withdraw collaboration tools or software producibility tools/technology from SWiT.
5	Register SWiT Instance	Enables a SWiT User to register their own SWiT infrastructure, making it available for others to use.
6	Deregister SWiT Instance	Enables a SWiT User to decommission their own SWiT infrastructure from being used by others.

Table 7. SWiT Infrastructure Management Use Case Summary

4.5 SWiT System Architecture

This section describes the SWiT system architecture. Architectural requirements are first presented, followed by the SWiT logical architecture. A proposed SWiT implementation architecture is then given. Finally, a brief overview of deployment objectives is presented.

4.5.1 Architecture Requirements

The primary design goals affecting the SWiT architecture are that SWiT must enable a high degree of collaboration around challenge problems, candidate solutions to challenge problems and experimentation related to challenge problems and candidate solutions, as well as support a vast range of experimentation with various software producibility technologies and requirements. A secondary design goal is that SWiT be flexible with respect to performance, capabilities and cost – e.g., be readily scalable to handle a larger number of users, greater security and more extensive experimental evaluations, as needed. Expanding upon these goals leads to the following requirements that must be met by the SWiT architecture:

R1. SWiT must enable rapid creation of complex, collaborative experiments. This includes supporting simultaneous execution of multiple, isolated experiments in a representative testbed.

R2. SWiT must support the ability to reserve and customize computer/network assets and configurations (to enable flexibility, scalability and security, for example).

R3. SWiT must provide the ability to save experiments and experimental contexts for later refinement/redeployment and continuous execution. For simplicity, SWiT experimentation should be web-accessible, where appropriate.

R4. SWiT must provide access to a repository of new technologies and both real and emulated system application components. This includes automated tool support for collaborating, designing and running experiments and analyzing experimental results.

4.5.2 Logical Architecture

To meet the requirements listed in the previous section, the SWiT system architecture (shown in Figure 9) is based on an e-commerce design that contains the following elements:

- **Web portal**, which receives requests from remote clients, processes the requests, and keeps track of the resources and user identities.

- **Services**, which support the various capabilities defined by the SWiT CONOPS, including account management, collaboration management, challenge problem management, candidate solution management and experimentation management. Tools to support/enable these capabilities are part of the Services element.

- **Database and file systems**, which house all of the data associated with SWiT, including web pages and forms for each of the major SWiT functions, and provide version control and issue tracking.

- **Testbed portal**, which will host the web pages and server pages that give status and permit configuration changes to a particular network accessible experimentation testbed.

- **Network accessible experimental testbeds**, which includes general-purpose processors, special-purpose processors, virtualization technology, and switches and routers.

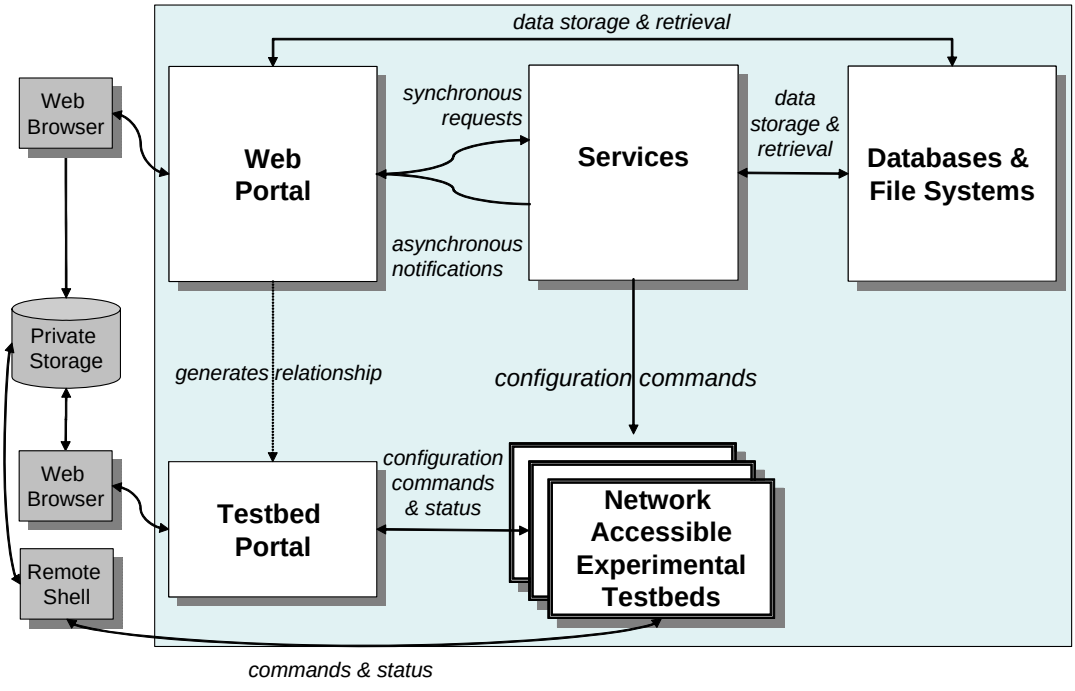


Figure 9. SWiT Logical System Architecture

4.5.3 Implementation Architecture

The SWiT logical system architecture has been designed so that the bulk of its elements can be developed using standards-based commercial-off-the-shelf (COTS) and/or open-source technologies. Specifically:

- The **web portal** consists of a web server (e.g., Apache or Microsoft IIS) to receive requests from remote clients, an application server (e.g., TomCat or JBoss) to process the requests, and a directory server (e.g., LDAP or Active Directory) to keep track of the resources and user identities managed by the web portal.

- SWiT **services** that support challenge problem management, candidate solution management, and some portions of experimentation management take the form of structured and free-form wikis. Account management is provided by, e.g., COTS identity management tools. Collaboration management is provided by, e.g., chat, messaging, talk, calendar/schedule management, Web Meeting, Mind mapping to link solutions with challenge problems, tools that capture human-to-human communication, and link to the testbed portal. Tools and technology provided as part of these services includes operating systems, compilers, licensed software (e.g., Simulink, Matlab) as well as experiment design tools (e.g., Emulab Experimentation Workbench, Skill, CUTS, etc.). Some of these tools and technology will, in fact, be provided by SISPI researchers.

- The **databases and file systems** consist of relational databases (e.g., MySQL, PostgreSQL, and Oracle), trees of HTML pages, forms, server pages for each of the major SWiT functions, artifact version management (e.g., Subversion and CVS), issue trackers (e.g., Bugzilla and Jira), web file system (e.g., WebFS), and text search (e.g., OpenFTS).

- The **testbed portal** mimics functionality provided by the Emulab Experimentation Workbench.

- The **network accessible experimental testbeds** include commercial technology (e.g., blades and blade centers, radiation hardened avionics mission computing processors, etc.), commercial virtualization technology (e.g., VMware, Frisbee, etc.), and commercial switches and routers (e.g., Cisco, Foundry, etc.).

Standards-based protocols, such as HTTP and TCP, can be used to interconnect many of the elements in the SWiT system architecture. Emulab protocols facilitate interaction between the various network accessible experimental testbeds, the testbed portal and SWiT services.

4.5.4 SWiT Operations Infrastructure

Figure 10 shows our vision for the long-term deployment of an implementation of the SWiT system architecture.

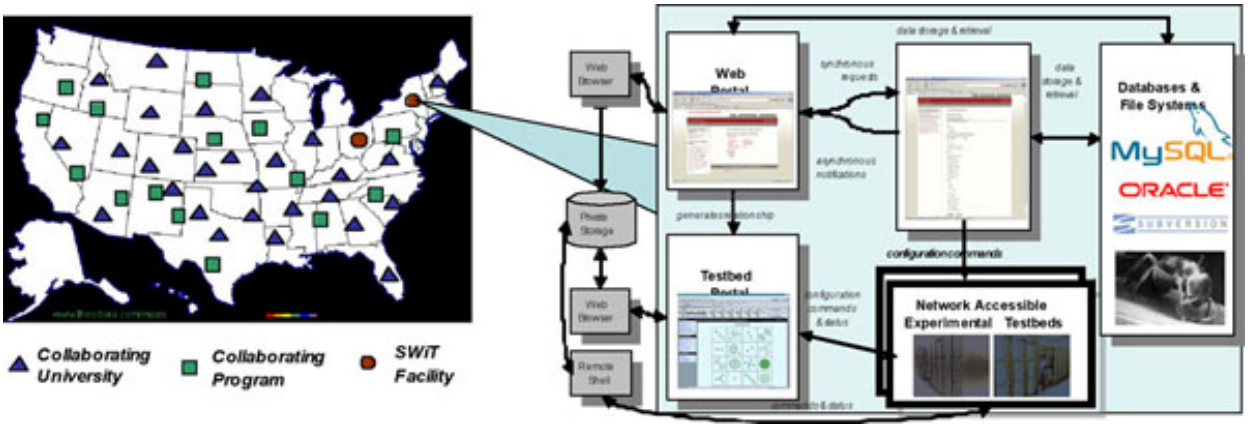


Figure 10. Initial SWiT Deployment

This deployment will consist of two Internet accessible facilities. These facilities could be housed at a Government Service Lab (e.g., AFRL) or a university or organization (e.g., the ESCHER Institute) that could provide secure—yet relatively open—access to SWiT Challenge Problem Providers and Candidate Solution Providers. Each facility could contain multi-processor or multi-blade servers to host full instances of the web portal, services and databases, and file systems described previously. The databases could be replicated across the two sites to ensure redundancy for fault tolerance and scalability.

Each SWiT facility could also contain testbed-specific resources, including racks of general-purpose blades, as well as racks of domain-specific hardware, such as embedded systems for avionics, shipboard computing, satellite computing. Each testbed would also provide at least 10 terabytes of RAID storage and firewall and domain name resolution (such as www.swit.mil or www.swit.org). In addition, licenses for commercial domain-specific tools would be included. We envision that there would be one system administrator per facility.

We propose to use largely open-source technologies as the basis of the initial SWiT deployment to reduce the development and licensing costs. Subsequent, larger-scale SWiT deployments could be based on a mix of open-source and commercial technologies to provide better scalability, security, and dependability.

Evaluating the feasibility, usability, and complexity of realizing the initial SWiT deployment will be guided by techniques and tools that our team has developed and/or applied over the past decade of research on AFRL, ONR, DARPA, and internally funded R&D (IRAD) programs. These technologies provide the starting points for realizing our capabilities, but will also be vetted thoroughly with needed extensions and enhancements identified based on feedback generated by our experience in Phase I of the SSTT program.

The SWiT testbed management capabilities will be built upon a network accessible, reconfigurable network-centric testbed with knowledge management tools for information sharing and collaboration. This testbed will provide hundreds of CPUs in racks connected by configurable high-speed routers and switches, combined with secure, user-friendly web-based tools, and driven by ns-compatible scripts and graphical user interfaces that will enable experimenters to remotely configure and control machines and links down to the hardware level. Packet loss, latency, bandwidth, queue sizes will all be user-defined and the operating system disk contents can be fully and securely replaced with custom images by experimenters.

4.6 SWiT Operational Policies and Constraints

Our intent is for SWiT to be an open resource, available for use by DoD program teams, SISPI Researchers, and SISPI Program Managers. To this end, we offer up the following issues and discuss what constraints they place on our system:

- Intellectual property protection
- DoD program classification
- ITAR restrictions

We now discuss each of these issues in turn.

4.6.1 Intellectual Property Protection

SISPI Researchers are often concerned about safe-guarding their ideas, at least until such time as they have proper protection for their intellectual property. SWiT, and the collaboration and experimentation process it enables, is designed to allow this intellectual property to be safe-guarded as desired by the SISPI Researcher – the SISPI Researcher is in full control over the level and quantity of intellectual property made available to others using SWiT.

Intellectual Property Protection of Challenge Problems

The SWiT policy is that only generalized/sanitized/non-proprietary challenge problems should be provided. Challenge problem providers are encouraged to omit proprietary problem details, both to protect intellectual property and to ensure that SWiT collaboration and experimentation activities do not become completely focused on short-term issues. Our policy is inspired by the Open Experimental Platforms (OEPs) activities on the DARPA MoBIES, PCES, NEST and ARMS programs, where sanitized challenge problems successfully guided research and technology development efforts.

We fully expect that researchers will create new software realizations and evolve existing software realizations of challenge problems as part of their experimentation.

SWiT also provides a means for challenge problem providers to restrict the set of SWiT users that can access challenge problem information.

Intellectual Property Protection of Candidate Solutions

The SWiT policy encourages providers of candidate solutions to challenge problems to document the results of their experiments and offer high level descriptions of their approaches, but to omit detailed technology descriptions (if desired, links to solution provider web sites can be provided). SWiT will permit candidate solution providers to upload their technology to SWiT for other SWiT users to download, but this is not a requirement.

Intellectual Property Protection of Experimentation

SWiT will grant experimenters exclusive access, including root privileges, to testbed resources reserved for their experiments. Experimenters are encouraged to remove all software installed on testbed resources once experiments are complete (SWiT will provide tools to make this a

single action). As part of the experimentation testbed resource reservation process, SWiT will provide a complete and clean install of a default operating system.

The above policies are inspired by the successful Emulab service, where multiple researchers conduct experiments using their private technologies.

4.6.2 DoD Classification

Numerous DoD programs involve proprietary activities that result in the program being assigned some form of DoD classification. SWiT is designed to operate at the UNCLASSIFIED level. This helps to encourage and promote open collaboration, and reflects our observation that software producibility challenges are typically orthogonal to operational requirements. As a result, SWiT challenge problems and candidate solutions will be unclassified, or can be sanitized to unclassified, while preserving the core challenges.

Software producibility technologies themselves are nearly exclusively unclassified. In fact, these technologies would ideally ultimately be transitioned to commercial off-the-shelf (COTS) vendors.

We expect that, with appropriate modifications (e.g., stronger authentication, logging, product restrictions, etc.), SWiT could be designed to operate at the SECRET level. SWiT is *not* designed to operate at multiple levels of security.

4.6.3 ITAR

SWiT is designed to operate with technologies that are not ITAR restricted. This matches our observations that software producibility challenges are typically *not* ITAR restricted, with the exception being when they are tightly coupled to computing/information technology that *is* ITAR restricted.

With modifications, we expect that SWiT could restrict access to U.S. persons:

- Requires verification of U.S. persons identity prior to granting the individual access to SWiT
- Requires strong labeling and logging of ITAR restricted and non-ITAR restricted technologies
- Requires training and explicit user acceptance of responsibility to protect ITAR restricted data

If ITAR is required, we may need to deploy SWiT in a closed environment.

4.7 SWiT Usage Scenarios

This section presents three technology transition usage scenarios for SWiT involving different levels of engagement of DoD program personnel and SISPI researchers for the purpose of evaluating SISPI research technology. While each usage scenario is possible (and may be necessary in specific instances), the usage scenario that we feel maximizes the potential for successful technology transition is, not unexpectedly, the one with greatest involvement of both DoD program personnel and SISPI researchers (this scenario is described in Section 4.7.3).

4.7.1 Candidate Solutions Evaluated on Open SWiT Testbed Only

In this usage scenario, illustrated in Figure 11, a DoD program defines sanitized challenge problems and SWiT Candidate Solution Providers run experiments in an *open* (i.e., network accessible) SWiT testbed. Results of the experimentation/evaluation are provided back to the DoD program.

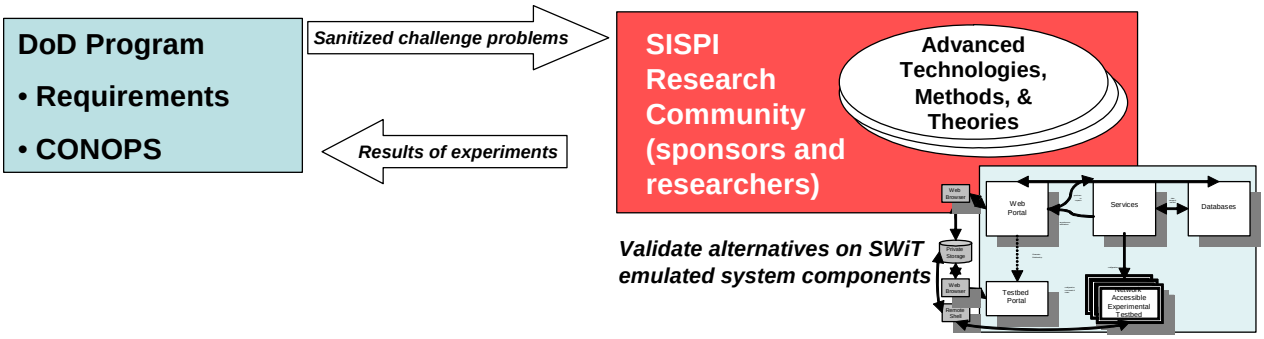


Figure 11. Evaluation on Open SWiT Testbed Only

This scenario offers up some key advantages in that minimal work needs to be done by DoD program engineers on evaluating the solution (i.e., the bulk of the work is done by the SISPI research community) while the SISPI research community benefits by getting an enhanced repository of sanitized challenge problems. However, we anticipate it will be difficult to transition technology artifacts because the SISPI research community only has access to limited statements of problems. Furthermore, results of SISPI research activities are unlikely to have significant impact because of a lack of serious commitment by the stakeholder (i.e., the DoD program).

4.7.2 Candidate Solutions Evaluated on Closed (Real) SWiT Testbed Only

In this usage scenario, illustrated in Figure 12, a DoD program uses a SWiT testbed in a *closed* environment to experiment with their own technologies. The SISPI research community acts only in a consulting role.

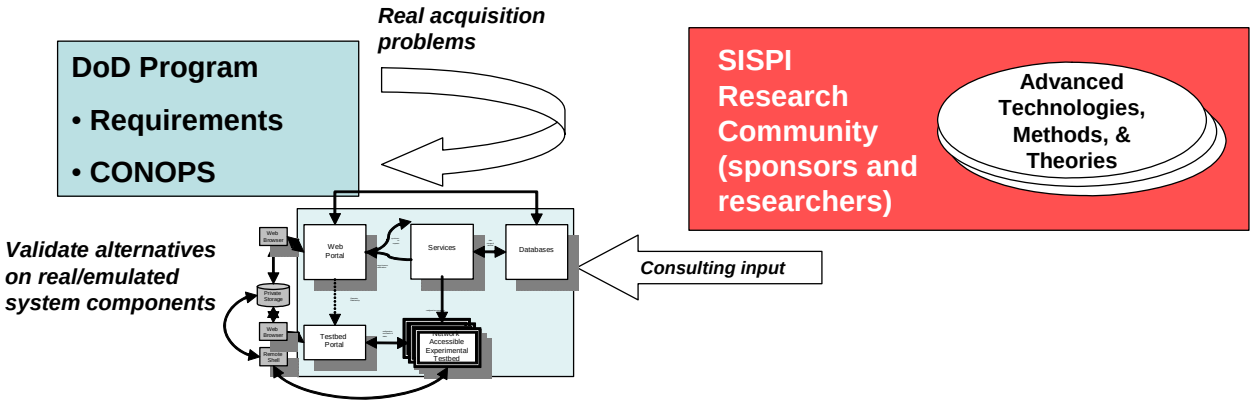


Figure 12. Evaluation on Closed (Real) SWiT Testbed Only

Advantages of this scenario favor the DoD program, in that the challenge problems and testbed environment are extremely high fidelity. Likewise, ITAR issues can easily be addressed. The

major disadvantage of this scenario is that the SISPI research community is relegated to a relative minor indirect role.

4.7.3 Candidate Solutions Evaluated on Open and Closed (Real) SWiT Testbeds

In this usage scenario, illustrated in Figure 13, a DoD program defines sanitized challenge problems and provides detailed specifications for the SISPI research community. SWiT Candidate Solution Providers implement their solutions to the specifications, run experiments in an *open* (i.e., network accessible) SWiT testbed to validate their candidate solutions, and provide prototype artifacts to Challenge Problem Providers (i.e., those on a DoD program) who run experiments in a *closed* SWiT testbed.

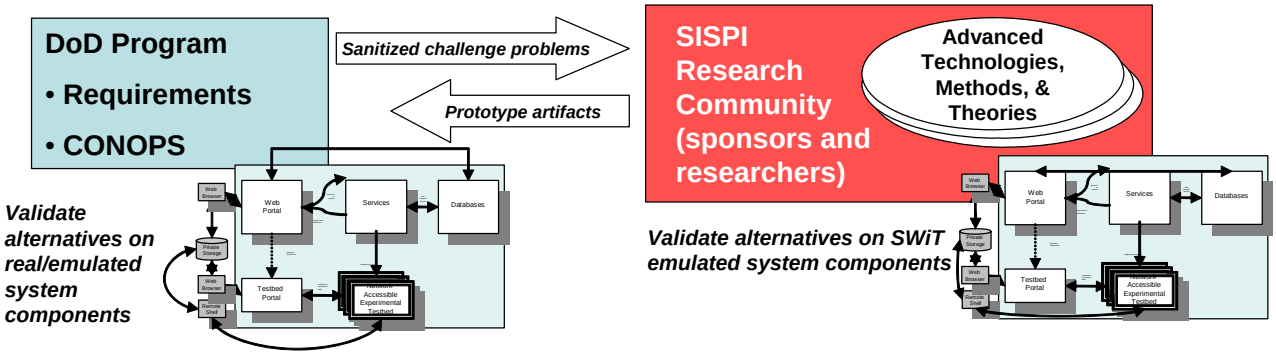


Figure 13. Evaluation on Open and Closed (Real) SWiT Testbeds

For this scenario, both DoD program engineers and SISPI researchers are involved in evaluations, so that the results are expected to be more meaningful. However, more commitment is required on part of the DoD program, which means that the SISPI research community must work harder on ensuring their activities are relevant.

Our belief is that this configuration appears the most likely to yield effective transition.

4.8 Prototyping and Validation Activities

4.8.1 Prototyping Activities

While the technical contributions of the contracted work effort called only for the development of a concept of operations and architecture to meet SSTT objectives, we elected to perform some prototyping activities to help us better understand how SWiT Users would interact with SWiT for aspects of SWiT operation. We developed partial initial prototypes of two components: a challenge problem description language and prototype web portals for challenge problem management. We discuss each of these prototyping activities in turn.

4.8.1.1 Challenge Problem Description Language

A “challenge problem description language” enables models of a broad array of challenge problems to be described in a structured way. A description can be prepared by researchers in order to describe the problems that can be solved by their existing technology. A description can also be prepared by DoD project personnel that need an easy way to begin describing the problems that they are facing, generally during the Concept Study phase of a project. Matches can then be made automatically through database cross-checking. Since the researchers and the

project personnel will be using the same problem description language, the odds of finding matches should be substantially improved over a simple free-form English prose approach to describing problems. An example of a language that we would recommend considering is expressed in EBNF below:

```
ChallengeProblem      ::= Issue “due to” (Cause)+ “. This results in” Impact
Issue                 ::= (“unable” | “difficult”) “to” Activity
                          (“on time” “with current resources” "under any circumstances")*
Impact                ::= (Issue)+ | (Severity Consequence)+
Activity              ::= (RequirementsActivity |
                           ArchitectureActivity |
                           DesignActivity |
                           ImplementationActivity |
                           TestActivity |
                           MaintenanceActivity)
Cause                 ::= “deficient” ("requirements" | “paradigm” | “process” |
                             "architecture" | "design" | "code" | "hardware" | “tools” |
                              “skills”)
Severity              ::= (“minor” | “major” | “catastrophic”)
Consequence           ::= (ProjectLevelConsequence |
                           DeploymentLevelConsequence)
ProjectLevelConsequence::= (“cost overrun” | “schedule slip” |
                          “reduced capability” | "risk to quality" | “project cancelled”)
DeploymentLevelConsequence::= (“system failure” | “system degradation”)
RequirementsActivity  ::= (“specify” | “modify” | “validate” |
                           “trace” | “reuse”) (“functional” |
                           “nonfunctional”) “requirements”
ArchitectureActivity  ::= (“specify” | “modify” | “validate” |
                           “trace” | “reuse” | “maintain integrity of”)
                           “architecture” (“model” | “document”)
DesignActivity        ::= (“specify” | “modify” | “validate” |
                           “trace” | “reuse”) “design” (“model” |
                           “document”)
ImplementationActivity ::= (“write” “review” “compile”
                           “configuration manage”) “source code”
TestActivity          ::= TestPlanningActivity | TestingActivity
TestPlanningActivity  ::= (“specify” | “modify” | “validate” |
                           “trace” | “reuse”) (“unit” | “integration” |
                          “system” | “customer acceptance”) “test plan”
TestingActivity           ::= (“develop testbed for” | “perform” |
                           “debug during” | “satisfy functional requirements during” |
                           “satisfy nonfunctional requirements during”)
                          (“unit” | “integration” | “system” |
                           “customer acceptance”) “testing”
MaintenanceActivity   ::= (“modify” | “upgrade” | “deploy”) “system”)
```

The following is an example of a testing problem description that can be specified in this language:

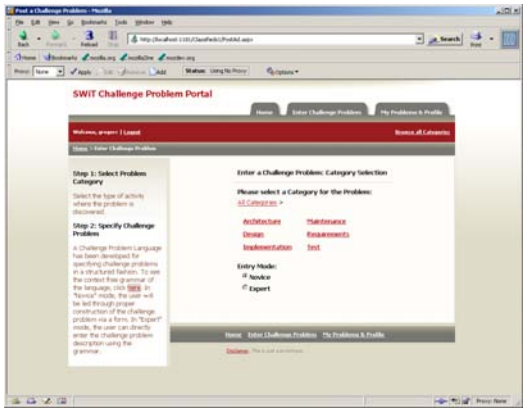
> *“Unable to satisfy nonfunctional requirements during system testing under any circumstances due to deficient tools. This results in major risk to quality.”*

We envision the SWiT Web Portal enabling the construction of a problem domain model database, starting with problem descriptions specified in a language like this. There could be an “expert” mode, where the user enters the description by simply typing in a description using this grammar. There could also be a “novice” mode, where the user was guided through the construction of a grammatically correct problem description.

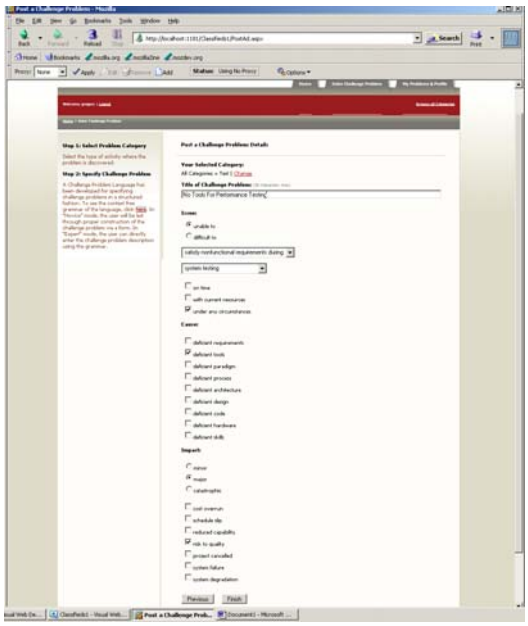
4.8.1.2 Prototype Web Portals for Challenge Problem Management

We developed several web portal screens associated with challenge problem management both to gain a better understanding of how SWiT might be used as well as to showcase initial ideas and solicit feedback from participants and consultants to SWiT activities. Specifically, we de-

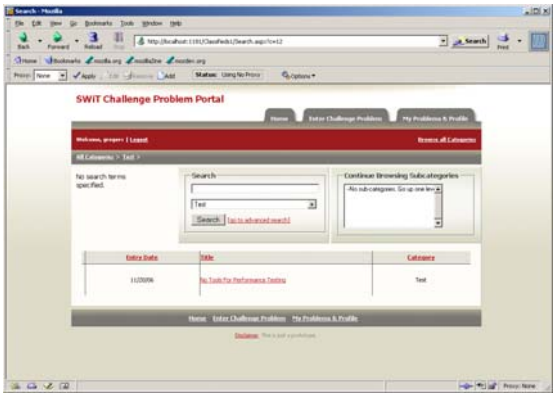
veloped prototype challenge problem portal components to enter challenge problems, post challenge problems, browse challenge problems, and view challenge problem descriptions. Each of the associated portal screens is illustrated in Figure 14.



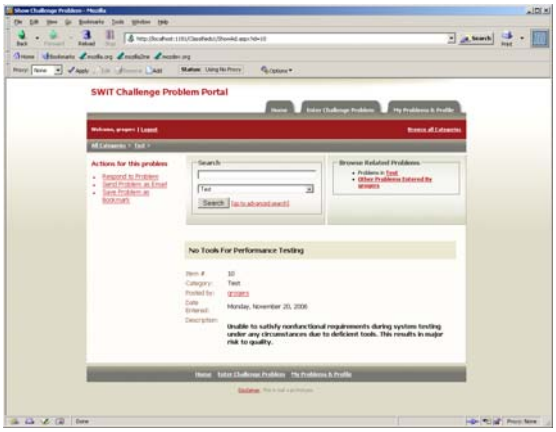
(a) SWiT Portal "Enter Challenge Problem" Screen



(b) SWiT Portal Screen "Post Challenge Problem Details"



(c) SWiT Portal Screen "Browse Challenge Problems in Test Category"



(d) SWiT Portal Screen "View Challenge Problem Description"

Figure 14. Sample SWiT Portal Challenge Problem Management Screens

4.8.2 Validation Activities

In addition to developing proof-of-concept prototypes, we validated SWiT CONOPS use cases by playing out an illustrative operational scenario that showcased the following activities:

- Creating a new challenge problem
- Collaborating in understanding and addressing the challenge problem

- Proposing a candidate solution to the new challenge problem
- Collaborating in understanding the proposed candidate solution to the new challenge problem
- Uploading and posting candidate solution technology into SWiT
- Creating and conducting experiments, and posting and analyzing experimental results, both to reproduce the new challenge problem in SWiT and to validate a specific candidate solution to the challenge problem
- Notifying the SWiT community that the DoD program engineer who created the new challenge problem intends to adopt the candidate solution to the challenge problem
- Having SWiT asynchronously notify registered participants of relevant events (e.g., when new challenge problems are posted, candidate solutions are posted, etc.).

In all of the above activities, SWiT records the interactions and experimental data, and maintains a detailed history of collaborations and experimentation. Such historical records will be useful to SISPI Program Managers as sources for new ideas and as a measure of program effectiveness to demonstrate the value of SWiT and the SSTT to sponsors.

5 Concluding Remarks

This report presented SWiT, an open collaborative research and development environment to demonstrate, evaluate, and document the ability of novel tools, methods, techniques, and runtime technologies to yield affordable and more predictable production of software intensive systems. DoD program engineers, SISPI researchers, DoD program managers and other personnel can define and collaborate around challenge problems, define and collaborate around candidate solutions to these challenge problems, and define, conduct and collaborate around experiments related to challenge problems and candidate solutions. While SWiT encapsulates several novel concepts, a key innovation of SWiT is its powerful, flexible, structured support for collaboration among DoD program engineers, SISPI researchers, DoD program managers and other personnel from the early stages of challenge problem definition through to late stage experimentation and validation of candidate solutions to these challenge problems leading up to eventual technology transition. This includes a seamless integration of the SWiT experimentation infrastructure with the SWiT collaboration/operations infrastructure. The concept of operations (CONOPS) for SWiT was discussed. The SWiT CONOPS was made necessarily broad in order to support the ability to transition technologies across the software producibility spectrum and across operational domains. Also presented in this report was a proposed SWiT architecture. A vast majority of the operational infrastructure components of the SWiT architecture and a sizeable portion of the experimental infrastructure components of the SWiT architecture can be realized with commercial off-the-shelf (COTS) or open source technologies. The SWiT CONOPS and architecture were developed, evolved and validated through input provided by a variety of DoD personnel, university researchers and DoD program engineers participating in a workshop and regular interactions with the SWiT team.

We envision that the use of SWiT will yield the following strategic benefits:

- SWiT will enable the incubation of promising prototype technologies that have proven successful in small-scale DoD capstone demonstrations.

- SWiT will support the maturation of advanced R&D technologies to validate their capabilities and encourage adoption by larger acquisition programs.

- SWiT will help create market demand to interest commercial vendors, open source development communities and standards bodies to ensure the transition of R&D technologies to off-the-shelf product offerings that can be applied successfully by DoD acquisition programs.

6 Recommendations

6.1 General Recommendations

Our team experiences, on this first phase of the Systems and Software Test Track program as well as other DoD programs, lead us to make the following general recommendations:

- Avoid trying to be all things to all programs initially. Focus on providing/supporting some very specific capabilities for a specific and restricted set of software producibility domains.

- Aiming for perfection may be misplaced in some cases. For example, some aspects of embedded systems are hard to emulate realistically. Aiming for perfect emulation may not be achievable and may not, in fact, even be necessary. Programs themselves often readjust the understanding of their programs over time. Furthermore, the Hawthorne effect is highly valuable in and of itself: a wide range of benefits come from just trying to model systems more realistically, irrespective of the fidelity of the proposed solutions. In many future-leaning DoD programs, existing practice is *extremely* limited, so there are many avenues for improvement.

- Problems with transition often seem less daunting when examined in a concrete context. Look for low-hanging fruit.

- We're not the only people facing problems with technology integration. E.g., consider "Airbus Vows Computers Will Speak Same Language After A380 Delay" [3]. Leverage the experiences of others and share our experiences with them.

The above general recommendations helped us to develop recommendations for follow-on/continued efforts in future phases of the Systems and Software Test Track program. These more focused recommendations are discussed next.

6.2 Operational Recommendations for SSTT Phase II and Beyond

Our overall operational recommendations for Phase II and beyond of the Systems and Software Test Track are:

- Realize a mini Systems and Software Test Track. Implement all CONOPS use cases. Construct a small testbed for initial experimentation (e.g., 10 nodes), based on general purpose computing hardware (to keep hardware costs low). Transition the infrastructure to a government-owned laboratory (e.g., AFRL). Provide a small number of support staff (e.g., one person) for testbed administration.

- Trial the Test Track implementation. Seed the Test Track with contrived collaborations using challenge problems with known existing solutions (provides a template for others to follow). Next, populate the Test Track with challenge problems for which candidate solutions have been proposed and are being developed. Third, populate the Test Track with challenge problems for which no candidate solutions have yet been proposed.

- Use feedback from Test Track trials to make Test Track improvements.

In the early stages, we recommend a focus on populating the challenge problem database. One option to help achieve this goal might be to hold an open challenge problem "competition" where a small number (e.g., 3-5) of small-valued (e.g., $50K-$100K) contracts are awarded to the defense industry to post challenge problems inspired by real problems. Sample challenge problems may be offered to guide contract awardees.

For long-term success, it is critical to ensure that the Systems and Software Test Track is self-sustaining. To this end, we recommend encouraging government agencies to post BAA challenge problems in the Test Track and encourage these agencies to request that contract awardees use the Test Track to assess their solutions. We recommend charging contract awardees for Test Track usage, and allow contract bidders to include costing to cover Test Track usage.

7 List of Symbols, Abbreviations and Acronyms

Table 8 defines all acronyms used within this document.

Acronym	Definition
ACC	Acquisition Community Connection
AFRL	Air Force Research Laboratory
BAA	Broad Agency Announcement
C2	Command and Control
C4ISR	Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance
CLAW	Control Law
CONOPS	Concept of Operations
DARPA	Defense Advanced Research Projects Agency
DAU	Defense Acquisition University

DoD	Department of Defense
FCA	Flight Control Application
ITAR	International Traffic in Arms Regulations
JSF	Joint Strike Fighter
LM ATL	Lockheed Martin Advanced Technologies Laboratory
NASA	National Aeronautics and Space Administration
NTTC	National Technology Transition Center
OEP	Open Experimental Platforms
SISPI	Software Intensive Systems Producibility Initiative
SSTT	Systems and Software Test Track
SWiT	Software Wind Tunnel
TRL	Technology Readiness Level

Table 8. Acronyms and Their Definitions

8 References

1. https://acc.dau.mil/CommunityBrowser.aspx?id=112228

2. http://www.dtic.mil/techtransit/about/overview.html

3. Rothman, Andrea, “Airbus Vows Computers Will Speak Same Language After A380 Delay” http://bloomberg.com/apps/news?pid=20601109&sid=aSGkIYVa9IZk&refer=%20news